Attorney Docket No.: 015114-066600US
Client Reference No.: A1052

PATENT APPLICATION

TECHNIQUES FOR MAPPING FUNCTIONS TO LOOKUP TABLES ON PROGRAMMABLE CIRCUITS

Inventor(s): Yean-Yow Hwang, a citizen of The United States, residing at
4231 Suzanne Drive
Palo Alto, CA 94306

Richard Yuan, a citizen of Peoples Republic of China, residing at
20199 Suisun Drive
Cupertino, CA 95014

Assignee: Altera Corporation
101 Innovation Drive
San Jose, CA, 95134

Entity: Large

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 650-326-2400

TECHNIQUES FOR MAPPING FUNCTIONS TO LOOKUP TABLES ON PROGRAMMABLE CIRCUITS

BACKGROUND OF THE INVENTION

[0001] The present invention relates to techniques for mapping functions to lookup tables on programmable circuits, and more particularly, to techniques for mapping functions to lookup tables that reduce the amount of circuitry needed to implement the functions.

[0002] A field programmable gate array (FPGA) is a programmable integrated circuit. Programmable integrated circuits also include programmable logic devices (PLDs), programmable logic arrays (PLAs), configurable logic arrays, etc. Many programmable integrated circuits are hybrids of FPGAs and ASICs (application specific integrated circuits).

[0003] FPGAs typically include programmable logic blocks, programmable routing resources, and programmable input/output (I/O) blocks. Each programmable logic block typically contain combinatorial components such as multiplexers and lookup tables as well as sequential components such as flip-flops.

[0004] Lookup tables are the basic logic blocks in many FPGAs today. A lookup table (LUT) includes memory cells that can store the truth tables of an arbitrary function. A LUT with k inputs can implement a function with k input variables. A LUT stores an output value corresponding to each set of input values. The pattern of output values describes the function.

[0005] A LUT receives a set of N digital input signals. The input signals are treated as a binary value with N bits. The binary value of the input signals are used as a memory address to access one of the LUT memory locations. A LUT outputs bits stored in the memory location. The memory location has an address corresponding to the binary value of the input signals.

[0006] A programmable integrated circuit is configured to implement a design provided by a user. Typically, the types of circuit elements in a user design do not correspond to the types of circuit elements in the FPGA, because users are not usually aware of the particular architecture of the FPGA. For example, the user typically does not implement the user design using LUTs.

[0007] Lookup tables and other circuit elements in the FPGA are selected to implement equivalent functions in the user design. This selection process involves synthesis and technology mapping. During synthesis, the user-made design is converted into a network of logic gates. During technology mapping, the logic gates are converted into logic blocks on the FPGA. Logic blocks are blocks on the FPGA that contain LUTs and registers.

[0008] Shannon expansion is a well known technique that maps a Boolean function to a set of lookup tables. However, as the number of input variables to the function increases, Shannon expansion maps the function to a disproportionately larger number of logic blocks.

[0009] For example, if each logic block on an FPGA has one 6-input LUT, Shannon expansion can be used to map a 7-input variable function to 3 logic blocks, a 10-input variable function to 21 logic blocks, and a 12-input variable function to 85 logic blocks.

[0010] Therefore, it would be desirable to reduce the number of lookup tables that are needed to implement functions on a programmable integrated circuit.

BRIEF SUMMARY OF THE INVENTION

[0011] The present invention provides techniques for mapping user functions in a user design to lookup tables on a programmable integrated circuit (IC). User functions within a user design are rewritten as a composition of smaller, decomposed functions using a decomposition technique. The decomposed functions typically have less input variables than the user function.

[0012] The present invention attempts to find decomposed functions that can fit into lookup table configurations available on the programmable IC. If such decomposed functions cannot be found, the input variables are rotated within the user function. The present invention then attempts to locate another set of decomposed functions that can fit into a lookup table configuration available on the programmable IC based on the rotated input variables.

[0013] Other objects, features, and advantages of the present invention will become apparent upon consideration of the following detailed description and the accompanying drawings, in which like reference designations represent like features throughout the figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Figure 1A illustrates an example of a programmable logic block architecture with three lookup tables;

[0015] Figures 1B-1D illustrate examples of logic blocks that contain lookup tables on a programmable integrated circuit;

[0016] Figures 2A-2B illustrate a set of LUTs that perform a decomposed function in a user design for a programmable circuit according to embodiments of the present invention;

[0017] Figure 2C is a flow chart that illustrates a process for mapping functions to lookup tables on programmable circuits according to an embodiment of the present invention;

[0018] Figure 2D illustrates how the input variables for the LUTs of Figure 2B can be rotated according to an embodiment of the present invention;

[0019] Figure 3 is a simplified block diagram of a programmable logic device that can implement embodiments of the present invention; and

[0020] Figure 4 is a block diagram of an electronic system that can implement embodiments of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0021] The present invention provides techniques for mapping functions in a user design to lookup tables (LUTs) on a programmable integrated circuit. A decomposition technique is used to rewrite a function within a user design as a composition of smaller functions that have less input variables. Decomposition functions are well known to those of skill in the art.

[0022] Many types of logic functions can be decomposed into smaller functions. However, some types of logic functions that can be mapped to LUTs using Shannon expansion cannot be mapped to LUTs using decomposition.

[0023] Let $f(X)$ denote a logic function $f(x_1, x_2, x_3 \ldots, x_n)$, where $X = (x_1, x_2, \ldots x_n)$. If function $f(X)$ can be represented by a function of $g(y_1(X_F), y_2(X_G), x_H)$, where x_H belongs to X, then function $f(X)$ can be decomposed into a composition of smaller functions. In decomposition function $g(y_1(X_F), y_2(X_G), x_H)$, y_1 and y_2 are performed by lookup tables F and G, respectively, and variable x_H is an input to lookup table H. The configuration of lookup tables F, G, and H is illustrates in Figure 1A.

[0024] Functions $y_1(X_F)$ and $y_2(X_G)$ are performed in a first stage of the decomposed function by LUTs F and G and function $g(y_1(X_F), y_2(X_G), x_H)$ is performed in a second stage of the decomposed function by LUT H as shown in Figure 1A. The decomposition function is split into two stages, because the outputs of functions $y_1(X_F)$ and $y_2(X_G)$ must be generated first before function g is performed.

[0025] Functions $y_1(X_F)$ and $y_2(X_G)$ receive a first subset of input variables x_1-x_n. X_F and X_G are subsets of x_1-x_n. Function $g(y_1(X_F), y_2(X_G), x_H)$ receives the output signals of functions $y_1(X_F)$ and $y_2(X_G)$ and input variable x_H.

[0026] According to another embodiment, if function $f(X)$ can be represented by a function of $g(y_1(B), y_2(B),\ldots y_t(B), x_{b+1}, \ldots, x_n)$, then function $f(X)$ can be decomposed into a composition of smaller functions using a disjoint functional decomposition technique, where B is a bound set that equals $(x_1, x_2, \ldots, x_b)$, and B is a subset of X. This equation can also be expressed as $f(X) = g(Y(B), X - B)$. If t = 1, it is called a simple disjoint decomposition. Functions $y_1(B), y_2(B),\ldots y_t(B)$ are performed in a first stage of the decomposed function, and function $g(y_1(B), y_2(B),\ldots y_t(B), x_{b+1}, \ldots, x_n)$ is performed in a second stage of the decomposed function.

[0027] Other types of decompositions are related to the disjoint decomposition. A non-disjoint decomposition of $f(X)$ is the case when a bound set of variables appears in the support of function g. If $S = (x_j, \ldots, x_b)$ for some $j > 1$, then a non-disjoint functional decomposition of $f(X)$ is $g(Y(B), S, X - B)$. Variables in the set S are the non-disjoint variables. When S is an empty set, the decomposition becomes disjoint.

[0028] A partially dependent decomposition can be used when the support for some encoding function is a strict subset of B, for example, when $B_m = (x_1, \ldots, x_m)$ which is a subset of B. If the support of some partially dependent encoding function contains only one variable, this encoding function can be replaced by the variable, and the decomposition becomes a non-disjoint decomposition.

[0029] Figures 1B-1D illustrate examples of configurable logic blocks for a programmable integrated circuit. The logic blocks contain one or more LUTs and registers. The LUTs perform logic functions that are needed to implement a user design. Figure 1B illustrates an example of a logic block that is configured to implement one six input LUT and generates one output signal.

[0030] Figure 1C illustrates an example of a logic block that is configured to implement two five input LUTs. The first 5-input LUT receives input variables x0-x4 and the second 5-input LUT receives input variables x3-x7. Input variables x3-x4 are shared inputs between the LUTs. The logic block has no more than 8 unique input variables in all. The logic block generates two unique output signals, one from each LUT.

[0031] Figure 1D illustrates an example of a logic block that is configured to implement two four input LUTs. The 2 LUTs receive 4 independent input variables, for a total of 8 unique inputs. The first 4-input LUT receives input variables x0-x3 and the second 4-input LUT receives x4-x7. Each 4-input LUT generates a unique output signal.

[0032] Figures 2A-2C illustrates how a decomposed function can be mapped to a set LUTs according to an embodiment of the present invention. Many types of Boolean and logic functions can be decomposed, as discussed above.

[0033] For example, the non-disjoint decomposition of a 9-input function f(X) can be expressed as h(x0, x1, x2, x3, g1, g2), where g1 and g2 are functions of (x4, x5, x6, x7, x8). Functions g1 and g2 are first stage functions, and function h is the second stage function. The decomposition function h is dependent on 9 unique input variables x0-x8. A non-disjoint decomposition such as h can be mapped into a set of lookup tables (LUTs) according to the techniques of the present invention.

[0034] Figure 2A illustrates an example of how a 9-input variable function such as f(X) can be mapped to only 3 LUTs. Figure 2A illustrates a logic block 201 configured to implement two 5 input LUTs R and S and a logic block 202 configured to implement one 6-input LUT T. The LUTs of Figure 2A can implement function f(X) by mapping the non-disjoint decomposition h(x0, x1, x2, x3, g1, g2) to LUTs R, S, and T.

[0035] Specifically, LUT R in Figure 2A is programmed to perform function g1(x4, x5, x6, x7, x8). LUT S is configured to implement function g2(x4, x5, x6, x7, x8). Both 5-input LUTs R and S receive the same five input variables x4-x8. The 6-input LUT T is configured to implement non-disjoint decomposition function h(x0, x1, x2, x3, g1, g2). LUT T receives 4 unique input variables x0-x3 and the output signals from LUTs R and S.

[0036] By using the decomposition technique, only 2 logic blocks and three LUTs R, S, and T can implement function f(X), which has 9 unique input variables x0-x8. If Shannon

expansion were used to map function f(X) to a set of LUTs, 13 logic blocks, each with a 6-input LUT, are needed to implement the function f(X).

[0037] Figure 2B illustrates an example of how a 12-input function can be mapped to only 3 LUTs. For example, the non-disjoint decomposition of a 12-input function J(X) can be expressed as k(x0, x1, x2, x3, m1, m2), where m1 is a function of (x4, x5, x6, x7, x8), and m2 is a function of (x4, x5, x9, x10, x11). A non-disjoint decomposition such as k can be mapped into the set of lookup tables (LUTs) shown in Figure 2B, according to an embodiment of the present invention.

[0038] In Figure 2B, logic block 211 is configured to implement two 5 input LUTs U and V, and logic block 212 is configured to implement one 6-input LUT W. LUT U receives input variables x4, x5, x9, x10, and x11. LUT V receives input variables x4, x5, x6, x7, and x8. Input variables x4 and x5 are common inputs between LUTs U and V.

[0039] In the example of Figure 2B, logic block 211 can only process a maximum of 8 unique input signals. Therefore, LUTs U and V share two of the same input variables x4 and x5. LUT U is programmed to implement the function m1. LUT V is programmed to implement the function m2.

[0040] Logic block 212 is configured to implement LUT W. LUT W is programmed to implement non-disjoint decomposition function k(x0, x1, x2, x3, m1, m2). LUT W receives input variables x0-x3 and the outputs of functions m1 and m2 from LUTs U and V.

[0041] By using the decomposition technique, only 2 logic blocks 211-212 and three LUTs U, V, and W can implement function J(X). If Shannon expansion were used to map function J(X) to a set of LUTs, 85 logic blocks, each with a 6-input LUT, are needed to implement the function J(X).

[0042] Thus, figures 2A and 2B illustrate examples of how decomposition techniques can greatly reduce the number of LUTs and logic blocks that are required to implement functions in a user design for a programmable integrated circuit. The decomposition techniques save resources on the programmable integrated circuit and increase the speed and routability of user designs.

[0043] Figure 2C is a flow chart that illustrates an example of how functions can be mapped to lookup tables on a programmable integrated circuit according to an embodiment of the present invention. The specific order of the steps in the process of Figure 2C is shown

as an example of the present invention and is not intended to limit the scope of the present invention.

[0044] During technology mapping, networks of logic gates in the user design are converted into logic blocks. The techniques of the present invention can modify the technology mapping process to increase logic efficiency and conserve resources on the programmable IC.

[0045] At step 251, a user function in a user design for a programmable integrated circuit (IC) is decomposed into a set of decomposed smaller functions using a decomposition technique, such as one of the decomposition techniques mentioned above. If the user function cannot successfully be decomposed into a set of smaller functions, then step 254 is implemented. Step 254 is discussed below.

[0046] If the user function can successfully be decomposed into a set of smaller functions, then step 253 is implemented. At step 253, a determination is made as to whether the decomposed smaller functions fit into one of the LUT configurations available in logic blocks on the programmable IC. In general, step 253 checks whether the decomposed smaller functions can fit into one of the LUT configurations allowed by a particular FPGA architecture.

[0047] For example, the logic block configuration of LUTs shown in Figures 2A-2B are examined to determine whether the decomposed functions fit into any of them. If the decomposed fit into one of the LUT configurations available on the FPGA, a set of logic blocks are selected, and LUTs in these logic blocks are configured to implement the decomposed functions.

[0048] The LUT configurations shown in Figures 1A-1D and 2A-2B are merely two examples of LUT configurations for a programmable IC. Programmable ICs can have many other types of LUT configurations, including LUTs that receive any suitable number of input variables.

[0049] If the decomposed functions do not fit one the LUTs configurations available in the FPGA architecture (step 253), the inputs variables of the user function are rotated at step 254. Rotating the input variables means that at least two of the input variables are rearranged.

[0050] Preferably, input variables are swapped between the first stage functions and the second stage function. For example, input variables x0-x11 of user function J(X) can be

rotated by swapping at two or more of the input variables x0-x11 between the first stage functions and the second stage function. If the first stage functions accept input variables x4-x11 and the second stage function accepts input variables x0-x3, then any one or more of input variables x4-x11 can be swapped with input variables x0-x3.

[0051] Figure 2D illustrates a specific example of how input variables can be rotated within a user function. Box 221 represents first stage 221 of a decomposed function, and box 222 represents second stage 222 of the decomposed function. During the first attempt at decomposition, variables x4-x11 are input into the first stage functions, and variables x0-x3 are input into the second stage function.

[0052] In Figure 2D, input variables x3 and x10 have been swapped between first stage 221 and second stage 222, and input variables x2 and x6 have been swapped between stages 221 and 222. This is merely one example of the many possible rotations that can be performed on the input variables of a user function.

[0053] At step 251, an attempt is made to decompose the user function into another set of smaller functions based on the rotated input variables. In the example of Figure 2D, an attempt is made to decompose user function J(X) into function r(s1, s2, x0, x1, x6, x10), where s1 and s2 are first stage decomposition functions that receive input variables x4, x5, x2, x7, x8, x9, x3, and x11. Function r is a second stage decomposed function that receives the outputs of functions s1 and s2 and input variables x0, x1, x6, and x10.

[0054] If the second decomposition is determined to be successful at step 252, step 253 is repeated to determine if the new set of decomposed functions can fit into one of the LUT configurations available on the programmable IC. If the decomposition is not successful at steps 252 or 253, the input variables of the user function are rotated again at step 254. Steps 251-254 are repeated until a LUT configuration is found that can implement the decomposed functions. Steps 251-254 can be repeated for other functions in the user design.

[0055] Once the user functions in the user design have been converted into LUTs on the programmable IC, the LUTs can be placed in logic blocks on the programmable integrated circuit at step 255 using well known placement techniques. The LUTs are then routed at step 256 by configuring programmable routing resources on the programmable integrated circuit using well known routing techniques.

[0056] In summary, the present invention provides techniques for mapping functions to lookup tables. The present invention attempts to decompose a user function in a user design for an FPGA into a set of decomposed functions using a decomposition technique. Decomposition greatly reduces the number of lookup tables that are needed to implement the function.

[0057] The present invention attempts to fit the decomposed functions into a LUT configuration on the FPGA. If the decomposed functions do not fit into one the LUT configurations on the FPGA, the input variables of the user function are rotated. The present invention then attempts to decompose the user function again based on the rotated input variables to locate a new set of decomposed functions that can fit into a LUT configuration. Many different Boolean and logic functions can be implemented by lookup tables using these techniques.

[0058] Figure 3 is a simplified partial block diagram of an exemplary high-density PLD 300 wherein techniques of the present invention can be utilized. PLD 300 includes a two-dimensional array of programmable logic array blocks (or LABs) 302 that are interconnected by a network of column and row interconnects of varying length and speed. LABs 302 include multiple (e.g., 10) logic elements (or LEs). A logic element is a type of logic block implements user defined logic functions.

[0059] PLD 300 also includes a distributed memory structure including RAM blocks of varying sizes provided throughout the array. The RAM blocks include, for example, 512 bit blocks 304, 4K blocks 306 and a MegaBlock 308 providing 512K bits of RAM. These memory blocks can also include shift registers and FIFO buffers. PLD 300 further includes digital signal processing (DSP) blocks 310 that can implement, for example, multipliers with add or subtract features. I/O elements (IOEs) 312 located, in this example, around the periphery of the device support numerous single-ended and differential I/O standards. It is to be understood that PLD 300 is described herein for illustrative purposes only and that the present invention can be implemented in many different types of PLDs, FPGAs, and the like.

[0060] While PLDs of the type shown in Figure 3 provide many of the resources required to implement system level solutions, the present invention can also benefit systems wherein a PLD is one of several components. Figure 4 shows a block diagram of an exemplary digital system 400, within which the present invention can be embodied. System 400 can be a programmed digital computer system, digital signal processing system, specialized digital

switching network, or other processing system. Moreover, such systems can be designed for a wide variety of applications such as telecommunications systems, automotive systems, control systems, consumer electronics, personal computers, Internet communications and networking, and others. Further, system 400 can be provided on a single board, on multiple boards, or within multiple enclosures.

[0061] System 400 includes a processing unit 402, a memory unit 404 and an I/O unit 406 interconnected together by one or more buses. According to this exemplary embodiment, a programmable logic device (PLD) 408 is embedded in processing unit 402. PLD 408 can serve many different purposes within the system in Figure 4. PLD 408 can, for example, be a logical building block of processing unit 402, supporting its internal and external operations. PLD 408 is programmed to implement the logical functions necessary to carry on its particular role in system operation. PLD 408 can be specially coupled to memory 404 through connection 410 and to I/O unit 406 through connection 412.

[0062] Processing unit 402 can direct data to an appropriate system component for processing or storage, execute a program stored in memory 404 or receive and transmit data via I/O unit 406, or other similar function. Processing unit 402 can be a central processing unit (CPU), microprocessor, floating point coprocessor, graphics coprocessor, hardware controller, microcontroller, programmable logic device programmed for use as a controller, network controller, and the like. Furthermore, in many embodiments, there is often no need for a CPU.

[0063] For example, instead of a CPU, one or more PLDs 408 can control the logical operations of the system. In an embodiment, PLD 408 acts as a reconfigurable processor, which can be reprogrammed as needed to handle a particular computing task. Alternately, programmable logic device 408 can itself include an embedded microprocessor. Memory unit 404 can be a random access memory (RAM), read only memory (ROM), fixed or flexible disk media, PC Card flash disk memory, tape, or any other storage means, or any combination of these storage means.

[0064] While the present invention has been described herein with reference to particular embodiments thereof, a latitude of modification, various changes, and substitutions are intended in the present invention. In some instances, features of the invention can be employed without a corresponding use of other features, without departing from the scope of the invention as set forth. Therefore, many modifications may be made to adapt a particular

configuration or method disclosed, without departing from the essential scope and spirit of the present invention. It is intended that the invention not be limited to the particular embodiment disclosed, but that the invention will include all embodiments and equivalents falling within the scope of the claims.